ScrollBar Pro Component

Technical Documentation

The ScrollBar Pro is a great component to be used when you need scrolling capabilities for large size elements inside small viewing areas. This component can be used for vertical or horizontal scrolling and can be customized as you wish, by changing the component's colors or even using your own clips as the elements composing the scroll bar. In order to use your own movie clips, they have to use the following linkage ids:

- "hsLeftButton" for the left button of the horizontal scroll bar
- "hsRightButton" for the right button of the horizontal scroll bar
- "hsScrollButton" for the scroll button of the horizontal scroll bar
- "hsScrollbarBackground" for the background of the horizontal scroll bar (the area on which the scroll button moves)
- "vsLeftButton" for the left button of the vertical scroll bar
- "vsRightButton" for the right button of the vertical scroll bar
- "vsScrollButton" for the scroll button of the vertical scroll bar
- "vsScrollbarBackground" for the background of the vertical scroll bar (the area on which the scroll button moves)

The component can also be customized by assigning the linkage ids to the corresponding parameters in the Component Inspector: forwardButton, backwardButton, scrollButton, scrollbarBackground.

ScrollBar Pro can be accessed from Components panel under UI Pro Components – jumpEYE.

Properties in the Parameters panel

horizontalScroll – sets the scrollbar for horizontal or vertical scrolling – default is “false”

target - the target clip that needs to be scrolled

Properties in the Component Inspector

keepScrollButtonSize - if "true", keeps the scroll button at the same size (the size from the library) regardless of the scroll area and the target clip size – default is “false”

backwardButton - the linkage id of clip from the library used as the component's up or left button (if you would like to customize the look of the component)

forwardButton - the linkage id of clip from the library used as the component's down or right button (if you would like to customize the look of the component)

scrollbarBackground - the linkage id of clip from the library used as the component's background (if you would like to customize the look of the component)

scrollButton - the linkage id of clip from the library used as the component's scroll button (if you would like to customize the look of the component)

Methods

setBounds(width:Number, height:Number) - sets the size of the scroll area in which the target clip will be scrolled; by default, the height of the scroll area is the same with the vertical scroll bar's height and the scroll area's width is the same with the vertical scroll bar's width
parameters:
- width:Number – the new width for the scroll area
- height:Number – the new height for the component

setSize(newWidth:Number, newHeight:Number) - sets the size of the scrollbar
parameters:
- newWidth:Number – the new width for the component
- newHeight:Number – the new height for the component

refresh() - redraws the scroll bar

updateScrollBarPos() - updates the position of the scroll button after the target clip has moved

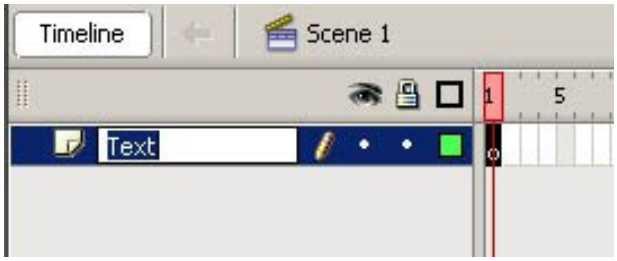
Notes

If you are attaching the scroll bars by ActionScript, the target clip should be the last parameter you set.
The default setting for the component is vertical scroll bar. In order to use it as a horizontal scroll bar you should modify the **horizontalScroll** property to "true" and resize the component to fit your needs.

IMPORTANT: Do not rotate the component to obtain a horizontal scroll bar, instead you should resize it to suit your needs.

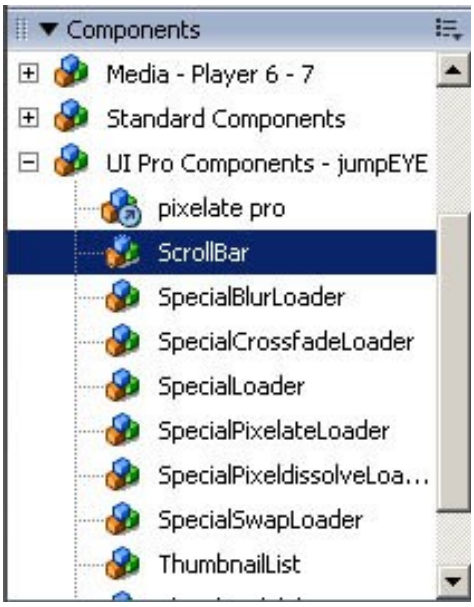
Setting up the scroll bars from the Flash IDE

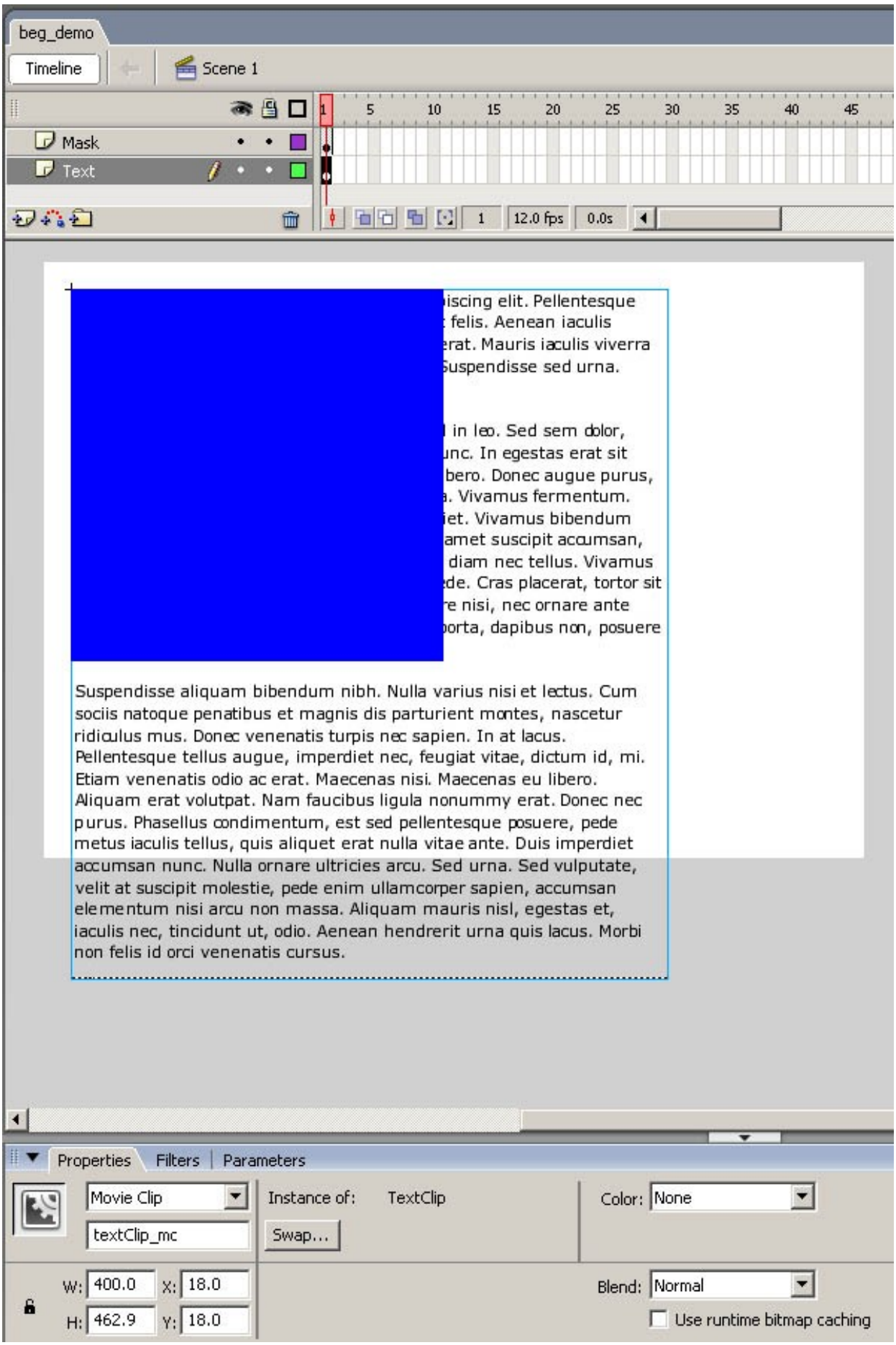
This tutorial will show you how to use this component without having to write any ActionScript code. This tutorial was intended to those who are new to Macromedia Flash. In this tutorial we'll be using the ScrollBar Pro component to scroll some text horizontally and vertically.
First you need to create a new .fla file by clicking on the File->New menu, select Flash Document and click on the Ok button. Now, give the document a name and save it by clicking on the File->Save As menu. Next, in the Timeline, double click on the first layer ("Layer 1") and change it's name to "Text". On this layer will reside the movie clip containing the text.



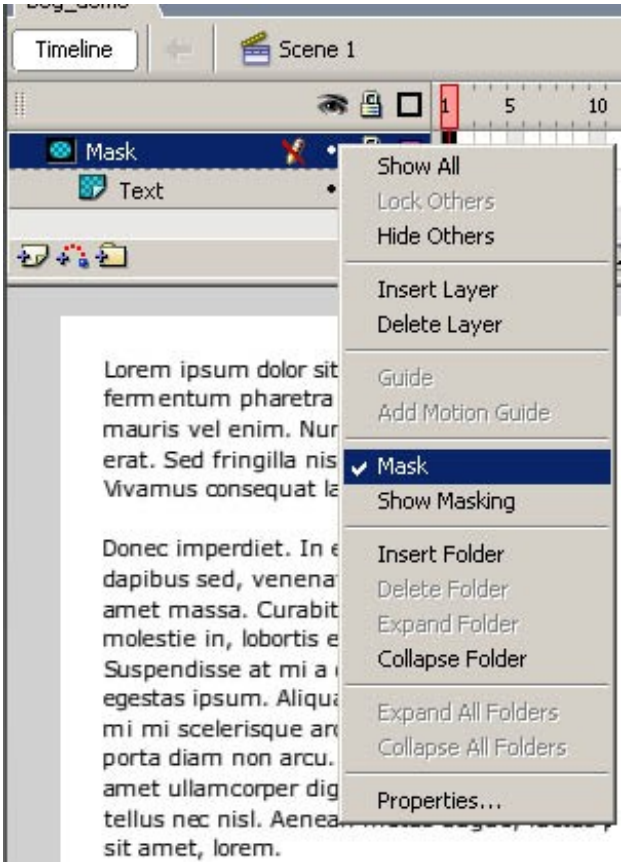
Now put a TextField on the stage, set it's width to 400 pixels and paste some large amount of text (a few paragraphs) from anywhere you like. Don't forget to set the TextField to "Multiline". If you set the text field to dynamic, you'll have to embed the characters used or you could leave it as static text field. Next, select the text field and press F8 to turn it into a movie clip. Give it the name of "TextClip" and an instance name of "textClip_mc".

In order to justify the usage of scroll bars, we have to display only a small portion of the text movie clip. This can be done by using a mask that will display the area underneath it and hide the rest. For this, you need to create a new layer and name it "Mask". On this layer you have to draw a rectangle shape, preferably without any stroke, and set the size to 250 by 250 pixels. Be sure to set the x and y coordinates of the rectangle to be the same as the text clip's coordinates. By now you should have two elements on the Stage (refer to the screen shot on the next page). Next, create a third layer and name it "Scroll bars". Here you need to drag two instances of the ScrollBar Pro component and place them at the edge of the shape found on the layer below. You can find the ScrollBar Pro component in the Components panel, "UI Pro Components – jumpEYE" folder.

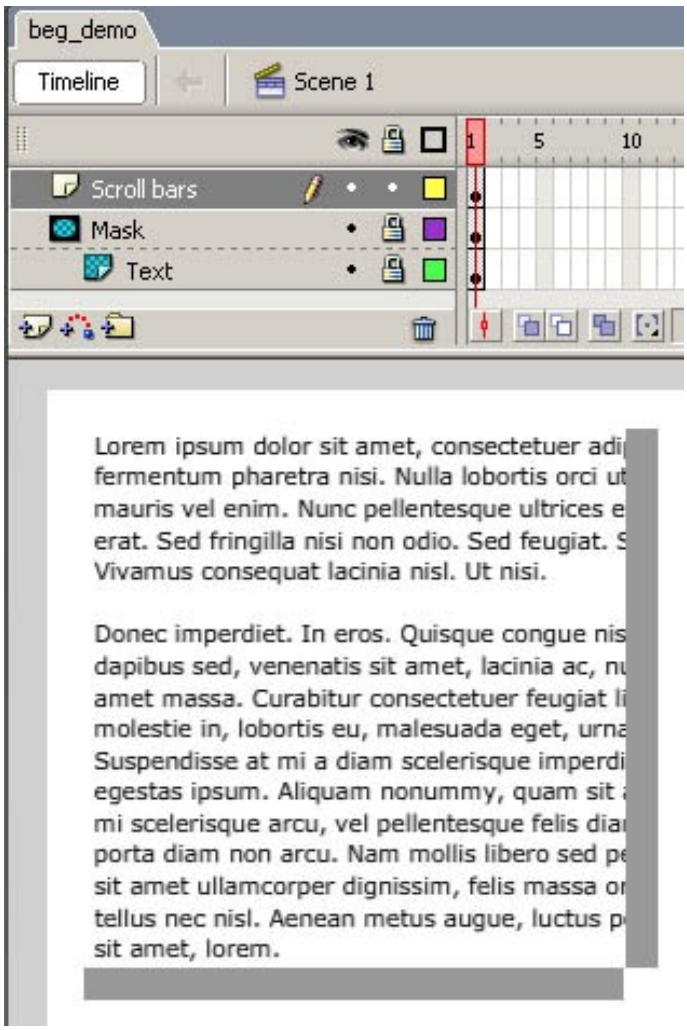


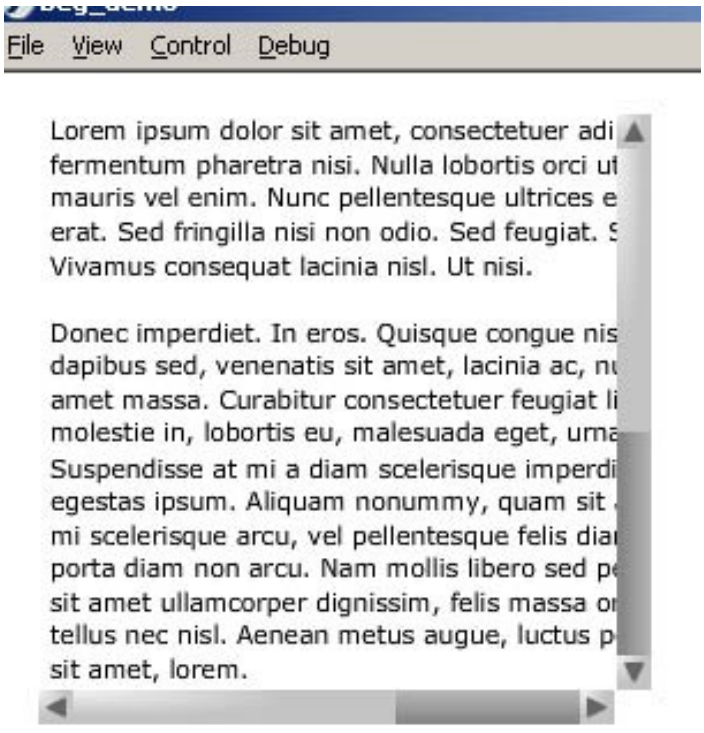

beg_demo
Timeline
Scene 1
Mask
Text
1
5
10
15
20
25
30
35
40
45
1
12.0 fps
0.0s
iscing elit. Pellentesque
felis. Aenean iaculis
erat. Mauris iaculis viverra
Suspendisse sed urna.
in leo. Sed sem dolor,
unc. In egestas erat sit
bero. Donec augue purus,
. Vivamus fermentum.
iet. Vivamus bibendum
amet suscipit accumsan,
diam nec tellus. Vivamus
de. Cras placerat, tortor sit
re nisi, nec ornare ante
porta, dapibus non, posuere
Suspendisse aliquam bibendum nibh. Nulla varius nisi et lectus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec venenatis turpis nec sapien. In at lacus. Pellentesque tellus augue, imperdiet nec, feugiat vitae, dictum id, mi. Etiam venenatis odio ac erat. Maecenas nisi. Maecenas eu libero. Aliquam erat volutpat. Nam faucibus ligula nonummy erat. Donec nec purus. Phasellus condimentum, est sed pellentesque posuere, pede metus iaculis tellus, quis aliquet erat nulla vitae ante. Duis imperdiet accumsan nunc. Nulla ornare ultricies arcu. Sed urna. Sed vulputate, velit at suscipit molestie, pede enim ullamcorper sapien, accumsan elementum nisi arcu non massa. Aliquam mauris nisl, egestas et, iaculis nec, tincidunt ut, odio. Aenean hendrerit urna quis lacus. Morbi non felis id orci venenatis cursus.
Properties
Filters
Parameters
Movie Clip
textClip_mc
Instance of:
TextClip
Swap...
Color:
None
W:
400.0
X:
18.0
H:
462.9
Y:
18.0
Blend:
Normal
Use runtime bitmap caching

Now, you'll have to set up the scroll bars: select the scroll bar you will use for vertical scrolling, set it's size to 15 x 250 pixels and set the **target** parameter to the text clip instance (textClip_mc). Now select the scroll bar you will use for horizontal scrolling set it's size to 250 x 15 pixels and the **target** to textClip_mc. For the horizontal scroll bar you have to set the **horizontalScroll** property to "true". Next you need to set the rectangle to be the mask for the text clip. You do that by right clicking on the layer named Mask, and, on the menu, select the Mask option.



Now you should have all the necessary elements set for the example to work properly (first image on next page). Next, you can test the example by pressing Ctrl+Enter. The second image on the next page shows you how the example should look like.


beg_demo
Timeline
Scene 1
1
5
10
Scroll bars
Mask
Text
Lorem ipsum dolor sit amet, consectetuer adi
fermentum pharetra nisi. Nulla lobortis orci ut
mauris vel enim. Nunc pellentesque ultrices e
erat. Sed fringilla nisi non odio. Sed feugiat. S
Vivamus consequat lacinia nisl. Ut nisi.
Donec imperdiet. In eros. Quisque congue nis
dapibus sed, venenatis sit amet, lacinia ac, nu
amet massa. Curabitur consectetuer feugiat li
molestie in, lobortis eu, malesuada eget, urna
Suspendisse at mi a diam scelerisque imperdi
egestas ipsum. Aliquam nonummy, quam sit
mi scelerisque arcu, vel pellentesque felis dia
porta diam non arcu. Nam mollis libero sed pe
sit amet ullamcorper dignissim, felis massa or
tellus nec nisl. Aenean metus augue, luctus p
sit amet, lorem.


File View Control Debug
Lorem ipsum dolor sit amet, consectetuer adi
fermentum pharetra nisi. Nulla lobortis orci ut
mauris vel enim. Nunc pellentesque ultrices e
erat. Sed fringilla nisi non odio. Sed feugiat. S
Vivamus consequat lacinia nisl. Ut nisi.
Donec imperdiet. In eros. Quisque congue nis
dapibus sed, venenatis sit amet, lacinia ac, nu
amet massa. Curabitur consectetuer feugiat li
molestie in, lobortis eu, malesuada eget, urna
Suspendisse at mi a diam scelerisque imperdi
egestas ipsum. Aliquam nonummy, quam sit
mi scelerisque arcu, vel pellentesque felis dia
porta diam non arcu. Nam mollis libero sed pe
sit amet ullamcorper dignissim, felis massa or
tellus nec nisl. Aenean metus augue, luctus p
sit amet, lorem.

Setting up the scroll bars using ActionScript

In this tutorial, I will show how to create the example from the previous paragraph with scroll bars added to the Stage using ActionScript. You should follow the previous example on how to create a new flash document and how to create the text field and the mask. Next, drag the ScrollBar Pro component from the Components panel to the Library. In the Timeline, crate a new layer, name it "Actions" and lock it, then press F9 to bring up the Actions panel. Insert the following code, to create the two scroll bars:

```
// create the two scroll bars
this.attachMovie("ScrollBar", "vertical_sc", this.getNextHighestDepth());
this.attachMovie("ScrollBar", "horizontal_sc", this.getNextHighestDepth());

// set the position and size for the two scroll bars
vertical_sc._x = 269;
vertical_sc._y = 18;
vertical_sc.setSize(15, 250);

horizontal_sc._x = 18;
horizontal_sc._y = 269;
horizontal_sc.setSize(250, 15);

// set the size of the scroll area
vertical_sc.setBounds(250, 250);
horizontal_sc.setBounds(250, 250);

// set the horizontal and vertical scroll bars
vertical_sc.horizontalScroll = false;    // this is default anyway
horizontal_sc.horizontalScroll = true;

// set the target clip to be scrolled
vertical_sc.target = textClip_mc;
horizontal_sc.target = textClip_mc;
```

The call to **setBounds()** method wasn't really necessary, since the size of the scroll area (the area in which the target clip is scrolled) is the same with the width of the horizontal scroll bar and the height of the vertical scroll bar. I used them only as an example of how to set the scroll area size using this method.

If you would like to update the position of the scroll button, according to the target clip's position (in case that clip can change it's position by other means than the scroll bar) you can use the **updateScrollButtonPos()** method each time the target clip has moved.

```
vertical_sc.updateScrollButtonPos();
```

If the size of the target clip changes, you can redraw the scroll bars using the **refresh()** method to resize the scroll button according to the target clip's new size. If the target clip's becomes smaller that the scroll area, the scroll button will disappear and the scroll bar will be disabled. If **keepScrollButtonSize** is set to **true**, the refresh method won't affect the size of the scroll button, since it will keep it's original size.

```
vertical_sc.refresh();
```

Customize the scroll bars

In order to customize the scroll bars from the previously created example, you have to bare in mind that the component will try to attach from the Library the composing elements of the scroll bar: the up and down buttons, the left and right buttons, the scroll button and the scroll area, the area on which the scroll button moves. If the movie clips in the Library have the correct linkage ids, the scroll bar will load those clips. Here are the movie clips I created for this example and their linkage ids:

- the up button of the vertical scroll bar (linkage id: "vsUpButton")

- the down button of the vertical scroll bar (linkage id: "vsDownButton")

- the scroll button of the vertical scroll bar (linkage id: "vsScrollButton")

- the scroll area of the vertical scroll bar (linkage id: "vsScrollbarBackground")

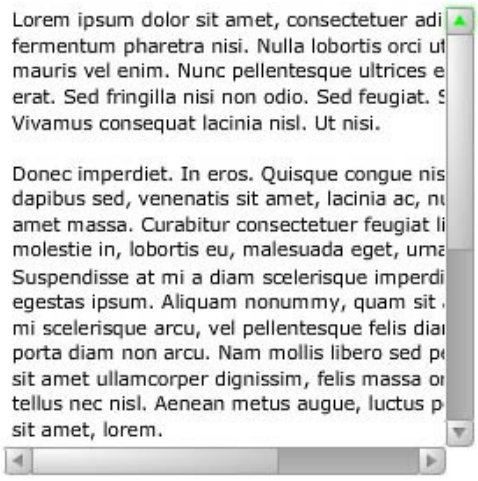
- the left button of the horizontal scroll bar (linkage id: "hsLeftButton")

- the right button of the horizontal scroll bar (linkage id: "hsRightButton")

- the scroll button of the horizontal scroll bar (linkage id: "hsScrollButton")

- the scroll area of the horizontal scroll bar (linkage id: "hsScrollbarBaclground")

The result would look like this:



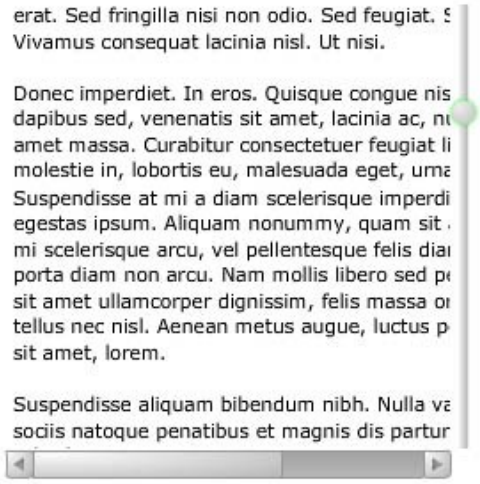
In this case, all the ScrollBar Pro instances will use these movie clips to construct the scroll bars. If you would like an instance of the ScrollBar Pro component to have a different set of movie clips, you could manually set the linkage ids to those movie clips in the Library. For example, I changed the skin of the vertical scroll bar with the next movie clips:

- for the up and down buttons I created two empty movie clips and gave them the linkage ids of "button1" and "button2"
- the scroll button of the second vertical scroll bar skin (linkage id: "scrollButton")

- the scroll area of the second vertical scroll bar skin (linkage id: "vertical")

After you have created the movie clips you need for the second vertical scroll bar skin, you'll have to assign them to the corresponding component properties. You'll find these properties in the Component Inspector or using ActionScript (**forwardButton, backwardButton, scrollButton, scrollbarBackground**). In order to keep the scroll button at the same size (regardless of the target movie clip's size and scroll area size, the component's **keepScrollButtonSize** property must be set to "true".

The result would look like this:



If you would like to set these properties using ActionScript, use the following code example:

```
vertical_sc.scrollbarBackground = "vertical";
vertical_sc.scrollButton = "scrollButton";
vertical_sc.forwardButton = "button2";
vertical_sc.backwardButton = "button1";
vertical_sc.keepScrollButtonSize = true;
```